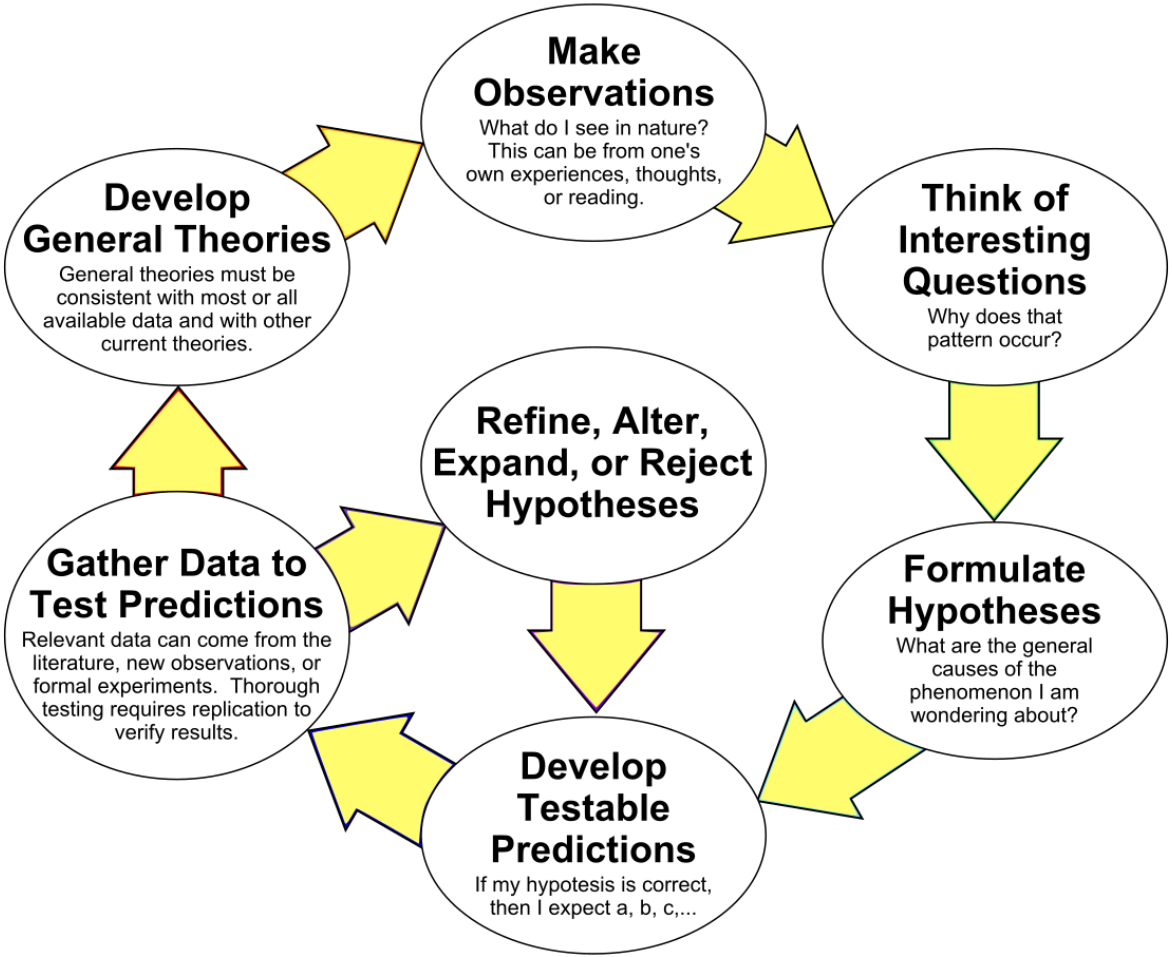
CYBERINFRASTRUCTURE
INTEGRATION RESEARCH CENTER
PERVASIVE TECHNOLOGY INSTITUTE

Assignment 1 Best practices checklist

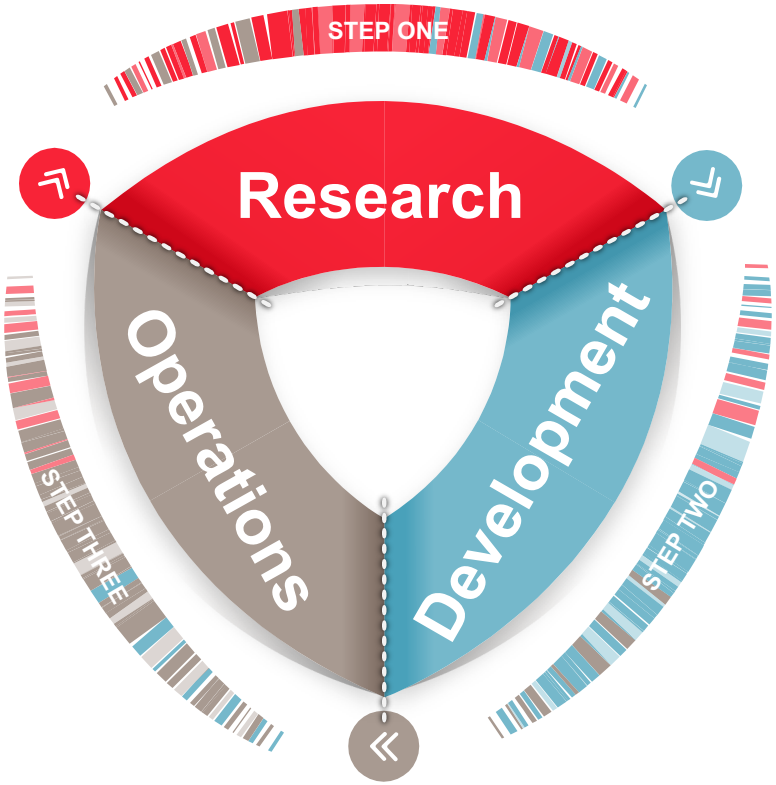
February 11th 2020

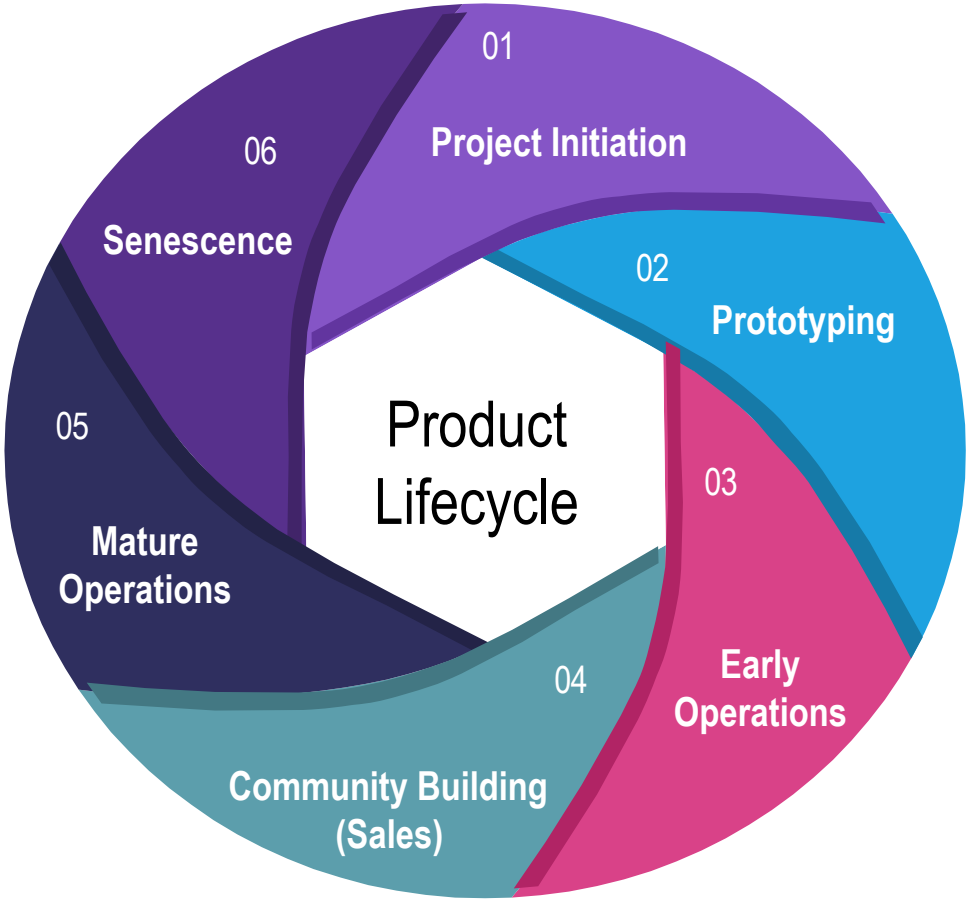
Suresh Marru & Marlon Pierce

The Scientific Method as an Ongoing Process

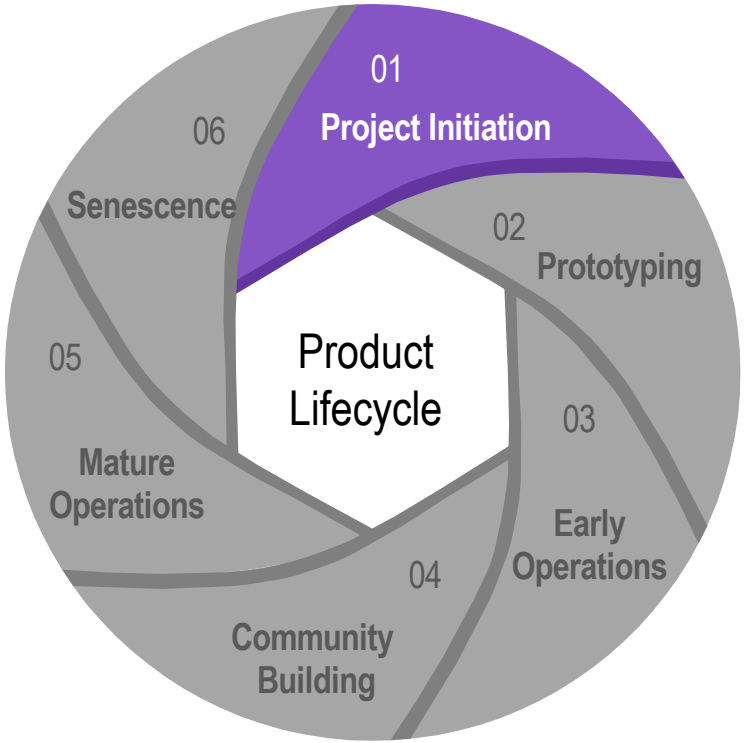


Lifecycle of typical Software System




Product Lifecycle
01
Project Initiation
02
Prototyping
03
Early Operations
04
Community Building (Sales)
05
Mature Operations
06
Senescence

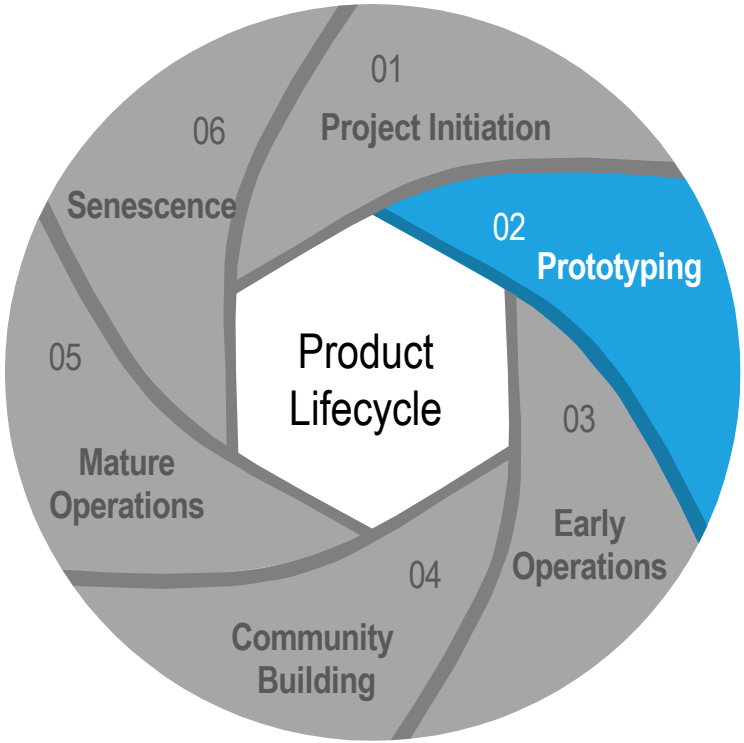
Project Initiation: you want to build a distributed system



Technical Challenges

- Establishing your technical base
- Choosing technologies that maps your value proposition.
- Assemble team: Who do you need

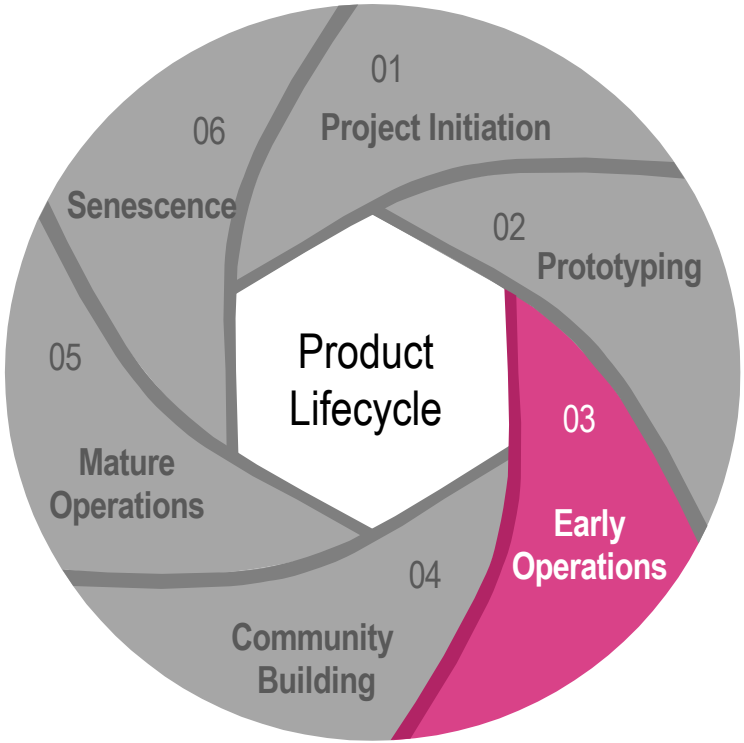
Prototyping:
Choosing a framework vs. building things yourself



Technical Challenges

- Evaluating Frameworks: How well can they be adapted to what you want?
- Assembling the right team, establishing the right engineering and operations practices
- Choosing third party systems

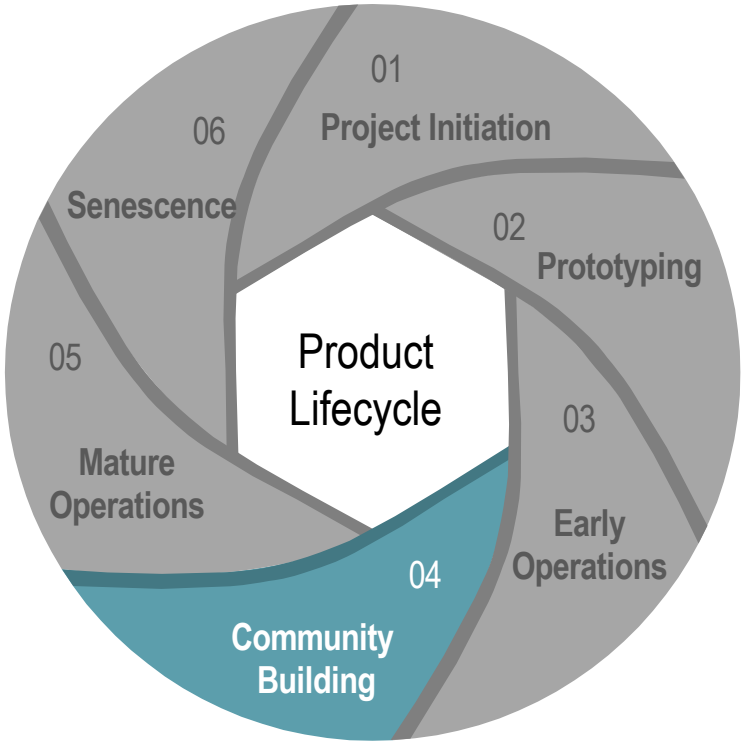
Early Operations:
Transitioning porotype into “production”



Technical Challenges

- High Availability
- Scaling up your technology for more users
- Monitoring
- Loss of key tech people; Onboarding new developers

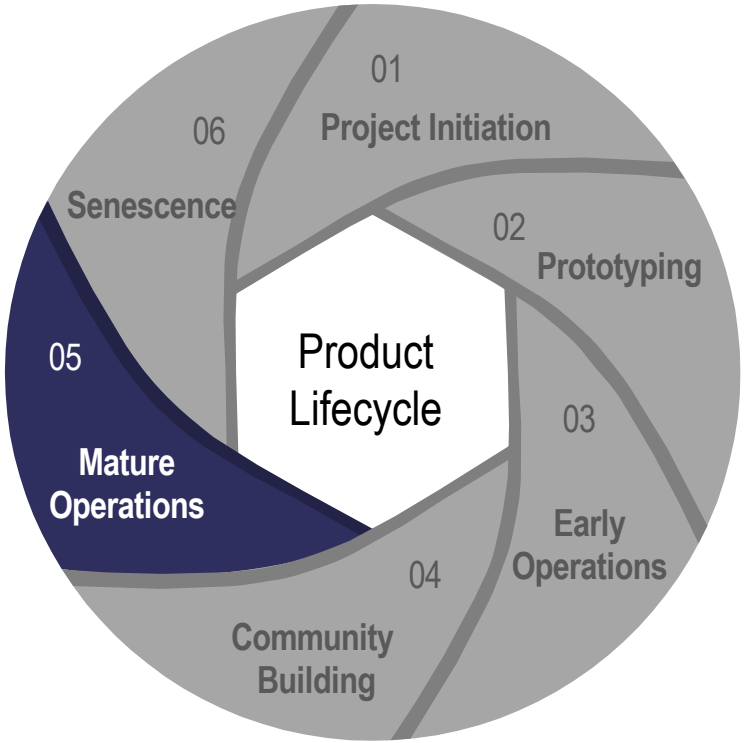
Community Building (Sales):
Have a technology but need to grow your user base



Technical Challenges

- Adding features expeditiously
- Managing feature creep

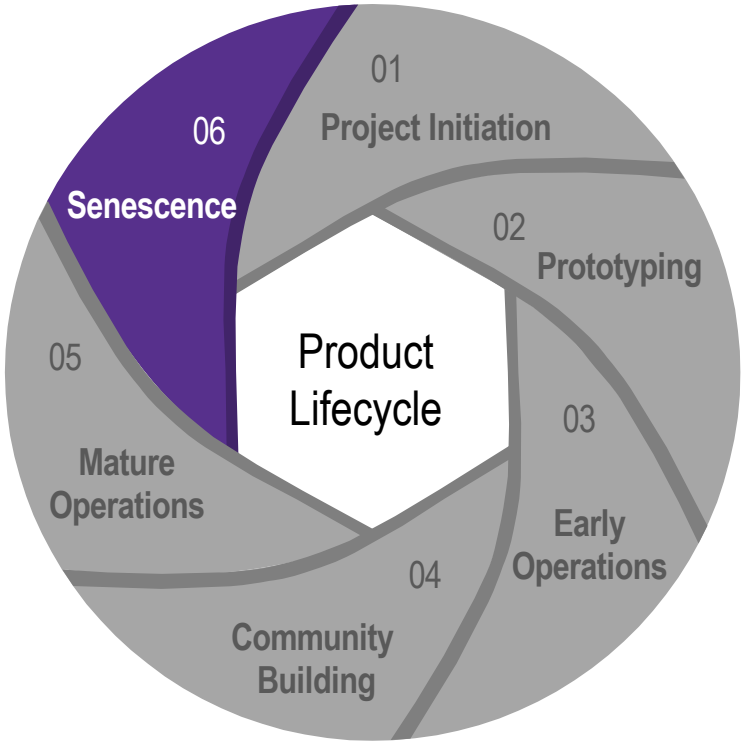
Mature Operations:
Mature in-house systems + User Community



Technical Challenges

- Technical Debt: dealing with earlier short cuts and shortcomings
- Expanding your team's roles
- Difficulties changing system to take advantage of new technology
- Loss of key people, onboarding
- Plateaued usage

Senescence:
Aging technology, value proposition may need revision



Technical Challenges

- Wholesale tech stack change
- Back to square one

Software Engineering in Context

- Think of SE from the point of view of this class and your architecture diagram. What do you need SE to do?
  - Enable you to expeditiously implement features and capabilities that are part of your value proposition.
  - Survive the departure of developers
  - Make new developers productive
- Good software engineering saves you time. A possible metric: number of commits on a key piece of code over time. Good code doesn’t need to be constantly changed.

Exercise (5 minutes)

Assume your project is going to lose your best teammate in 2 weeks (that may be you).

What software engineering practices will help your project survive the transition?

Some Answers

- Have a system architecture that matches your implementation
- Documented code
  - Issue tracking tied to versioning, commits
  - Strive for self-documenting code
- Code reviews:
  - More than one person has looked at the code
  - Comments captured
  - Avoid unnecessary complexity
  - Avoid implementations that don’t match the architecture
- Make sure version control is using a sensible branching strategy
- Have a build and test framework
  - At least one branch always builds
  - You have unit testing that covers all of your code
- Have a continuous integration system

Software Engineering Best Practice Checklist

1. We have our code in a version control system that meets our needs.
2. We associate all commits with issues.
3. We have at least two main branches for all of our code: develop and release.
4. We have code reviews for all code committed to our main branches.
5. We have a build system that we regularly execute.
6. Our build system includes unit tests.
7. We fix broken builds, including test failures as well as compilation failures.
8. We use code analysis tools to help find problems such as DRY code, code that is never used, code that is too complex, inadequate unit test coverage, etc.
9. We remove obsolete code.
10. Our integration and deployment scripts are in our code base.

Characteristics of a Good Technology Base

- ✓You are continually improving your code base
- ✓You are strategically adding major new capabilities
- ✓You get improvements expeditiously into production
- ✓You can replace key personnel
- ✓You get meaningful contributions
- ✓You have boring operations: the system as a whole doesn't break, security upgrades aren't a major hassle, etc.
- ✓Parts of your base get reused in other projects.

You are really focused on implementing your value proposition.

What Are Some Signs that Your Technology Base Needs Improving?

- You implement stuff that is not central to your value proposition that you could just get off the shelf
- You implement features and capabilities that aren't central to your value proposition.
- You have a lot of legacy features that you have to maintain
- You must deploy too many things manually
- Merging important new features into your production code takes too long

You spend too much effort on things that are not part of your value proposition